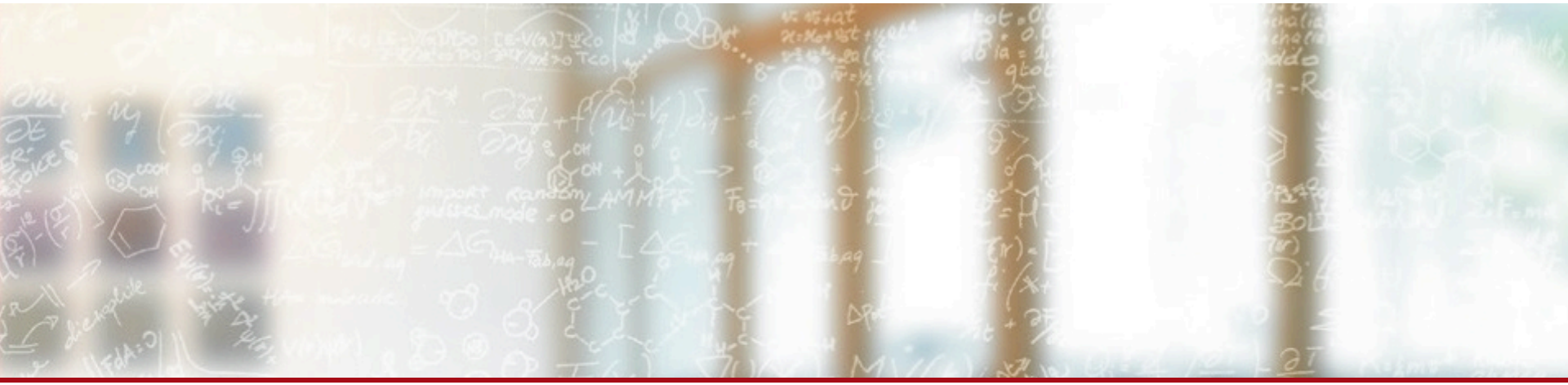
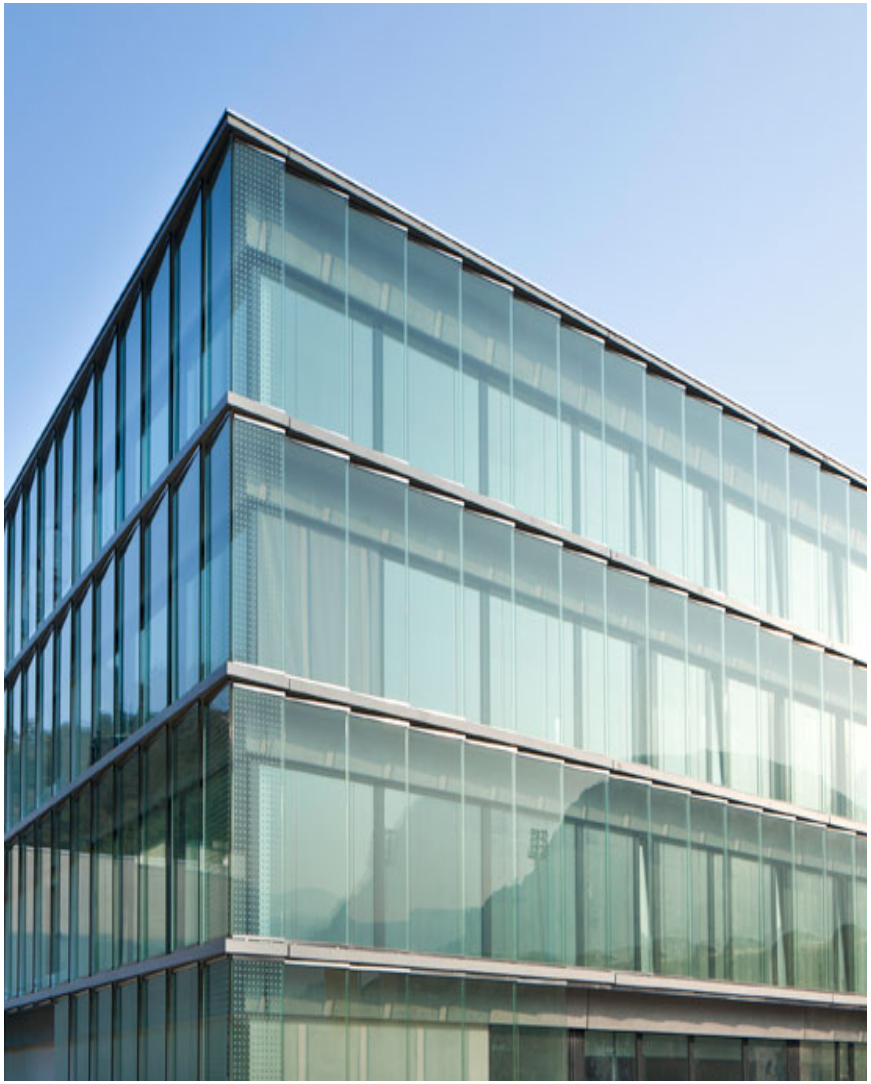


EasyBuild site talk: CSCS

9th EasyBuild User Meeting
Apr 23rd – 25th 2024, Umeå, Sweden

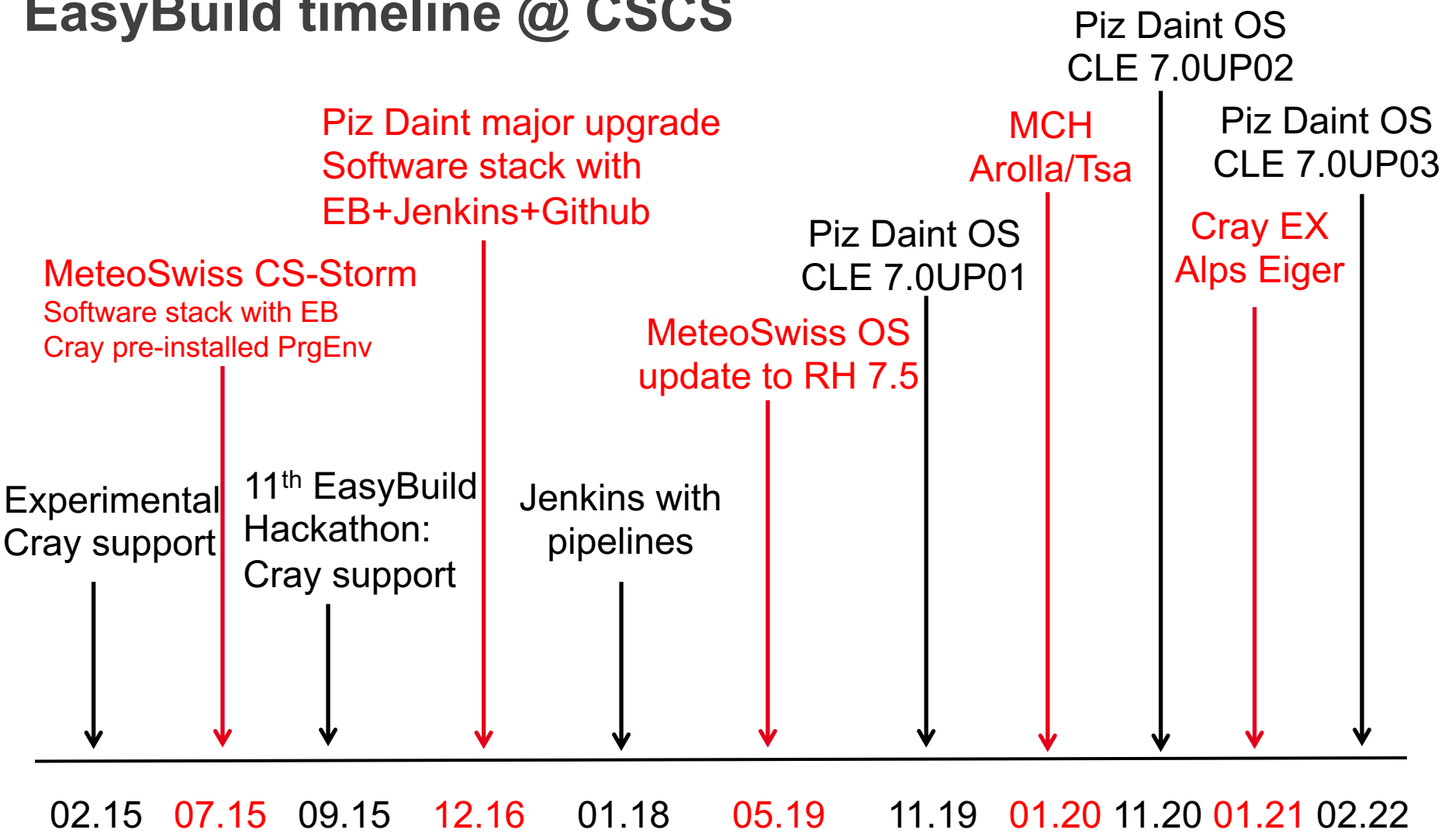
Luca Marsella
Swiss National Supercomputing Center (CSCS / ETHZ)

Outline

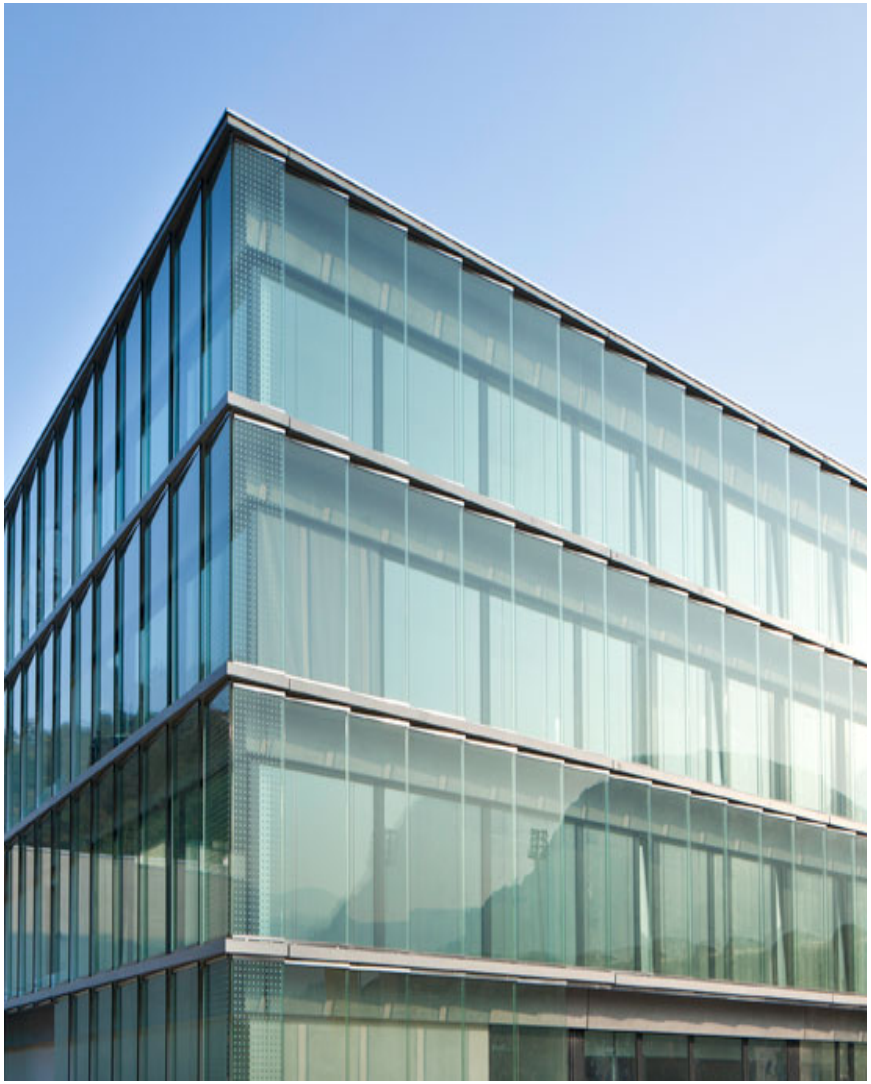


- **EasyBuild timeline @ CSCS**
- CSCS HPC systems
  - Piz Daint
  - Alps
  - MeteoSwiss systems
- EasyBuild for CSCS Users
  - Custom User builds
  - Jenkins pipelines
- UENV User Environments
  - CLI and Slurm integration
  - The Stackinator tool

EasyBuild timeline @ CSCS



Outline



- EasyBuild timeline @ CSCS
- **CSCS HPC systems**
  - Piz Daint
  - Alps
  - MeteoSwiss systems
- EasyBuild for CSCS Users
  - Custom User builds
  - Jenkins pipelines
- UENV User Environments
  - CLI and Slurm integration
  - The Stackinator tool

CSCS HPC systems

System	Users	Accelerators	Architecture
Piz Daint	HPC Platform	1 GPU	Cray XC50 / XC40 P100,Haswell/Broadwell
Alps Eiger	HPC Platform	CPU only	Cray EX AMD Rome
Alps Clariden	AI/ML Platform	4/8 GPU	Cray EX AMD Milan, A100/Mi200
Arolla / Tsa	MeteoSwiss	8 GPU	V100, Intel SkyLake

Piz Daint

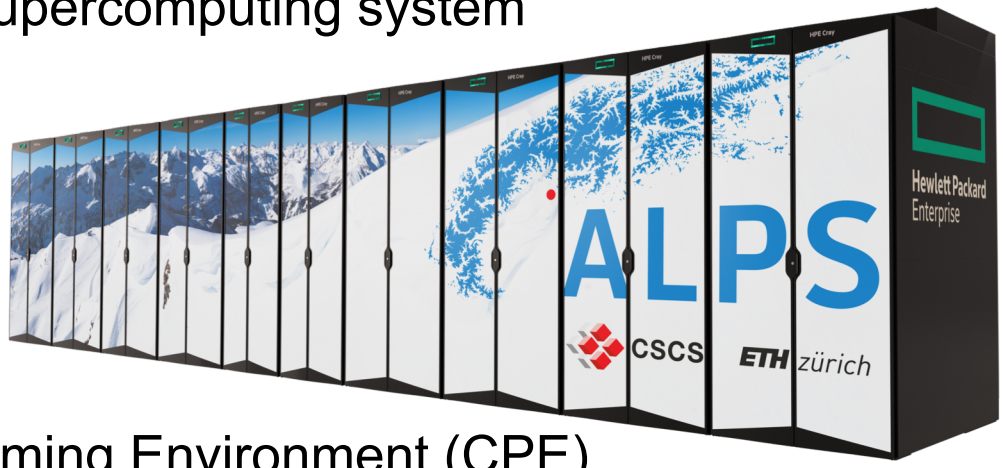
Model	Cray XC50 / XC40
XC50 node	Intel® Xeon® E5-2690 v3 (Haswell) @ 2.60GHz (12 cores, 64GB RAM) and NVIDIA® Tesla® P100 16GB
XC40 node	Intel® Xeon® E5-2695 v4 (Broadwell) @ 2.10GHz (18 cores, 64/128 GB RAM)
Login node	Intel® Xeon® CPU E5-2650 v3 @ 2.30GHz (10 cores, 256 GB RAM)
Interconnect	Aries routing and communications ASIC, and Dragonfly network topology
Scratch	8.8 PB (Lustre / Sonexion 3000)

Flagship production system with hybrid nodes

- EB software stack in production since 11.16
- Successfully updated OS to CLE 7.0UP03 in 02.22
- Automated update of Easyconfig files in production

Alps Eiger

Eiger on the **Alps** HPE/Cray EX Supercomputing system



Compute nodes
2 x AMD EPYC™ 7742 64-Core
HPE Slingshot interconnect

EB Toolchains
- Toolchains for the Cray Programming Environment (CPE)
  - cpeAMD, cpeCCE, cpeGNU, cpeIntel
- EB Hierachical Module Naming Scheme

EB Software stack
- Amber, GROMACS, LAMMPS, NAMD
- CP2K, QuantumESPRESSO, VASP
- ParaView, VisIt

MeteoSwiss systems

Arolla and **Tsa**
- Intel Skylake and Tesla V100
- EB software stack available since 01.20

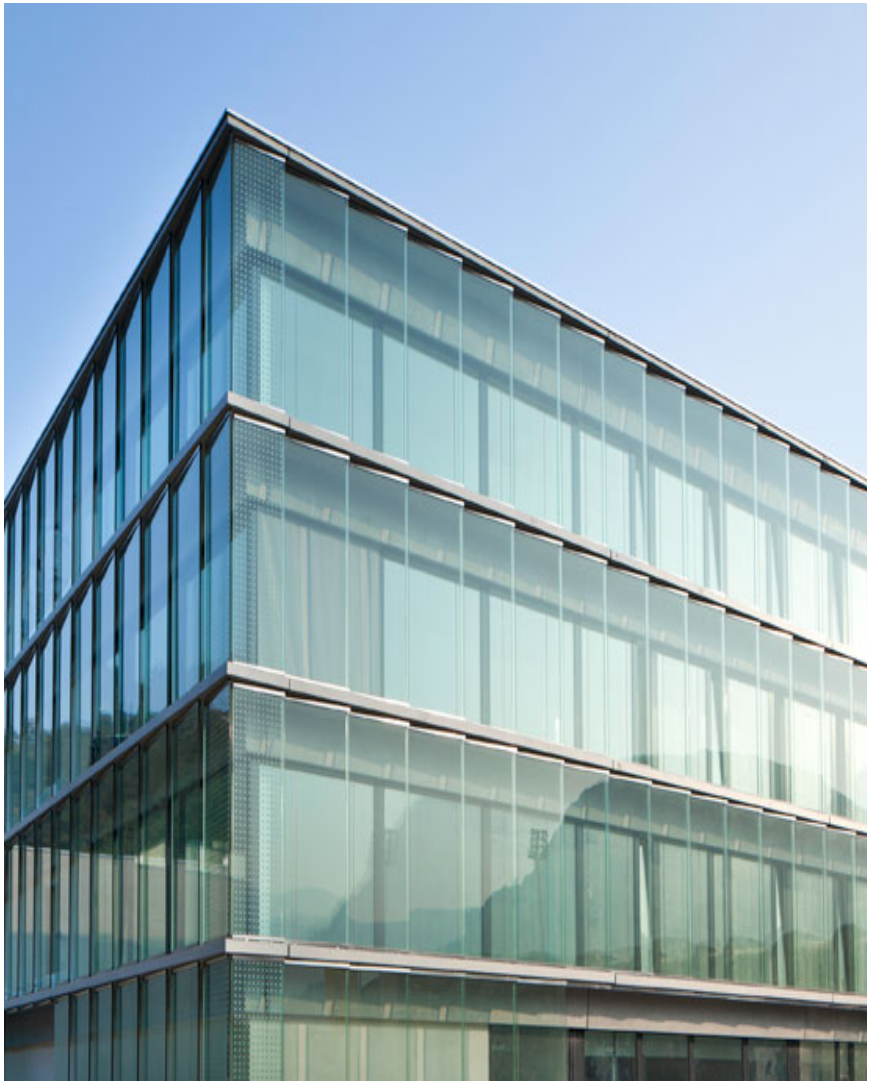
Lowercase module names (few exceptions)
- EasyBuild-custom (CSCS EB modulefile)
- PrgEnv-gnu
- PrgEnv-pgi

Meta-modules for hierarchical environment

PrgEnv-pgi/20.4 unfolds additional modules:
- hdf5/1.10.5-pgi-20.4-gcc-8.3.0
- netcdf-c++/4.3.0-pgi-20.4-gcc-8.3.0
- netcdf-fortran/4.4.5-pgi-20.4-gcc-8.3.0
- netcdf/4.7.0-pgi-20.4-gcc-8.3.0

…

Outline



- EasyBuild timeline @ CSCS
- CSCS HPC systems
  - Piz Daint
  - Alps
  - MeteoSwiss systems
- **EasyBuild for CSCS Users**
  - Custom User builds
  - Jenkins pipelines
- UENV User Environments
  - CLI and Slurm integration
  - The Stackinator tool

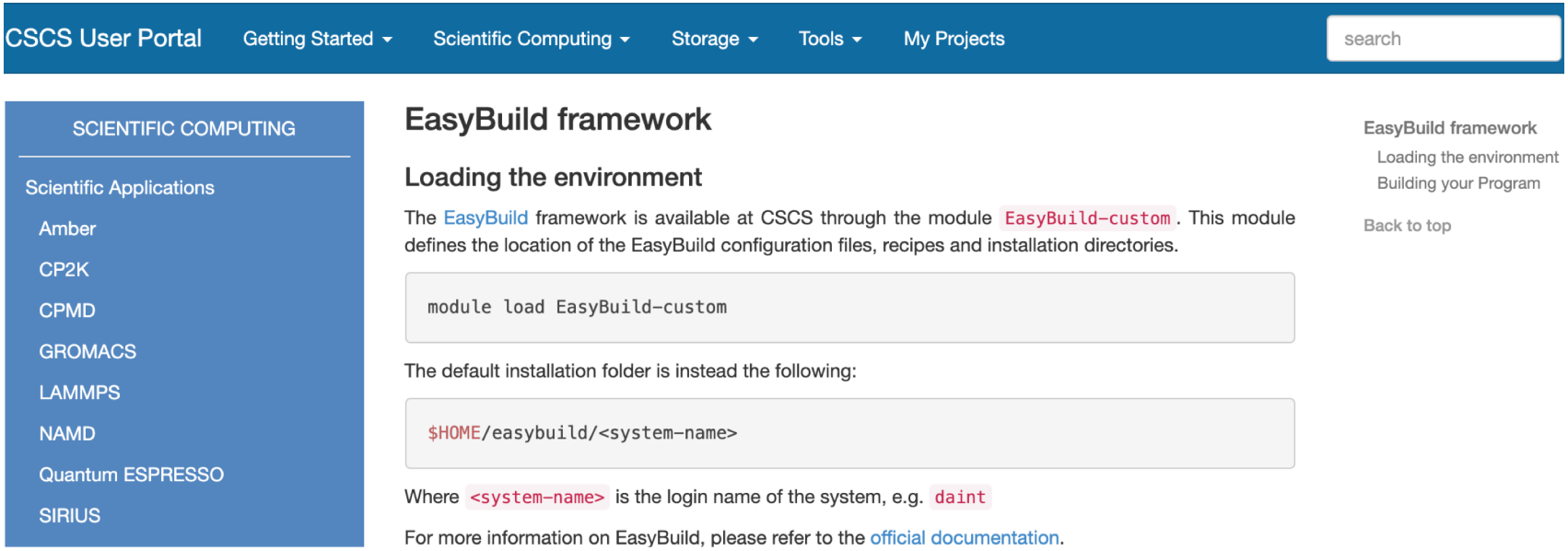
EasyBuild for CSCS Users

- EasyBuild recipes provided for software requests
  - Instead of error-prone manual steps on how to build and run

- EasyBuild documentation on the CSCS User Portal

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Custom User builds

- Users can extend or customize CSCS EasyBuild recipes
  - **git clone https://github.com/eth-cscs/production.git**
- Export the EB custom environment variable
  - **EB_CUSTOM_REPOSITORY=/<path>/production/easybuild**
- Load the EB custom modulefile
  - **module load EasyBuild-custom/cscs**

The modulefile **EasyBuild-custom/cscs** adds CSCS production Easyconfigs to the local robot path for search

EasyBuild with Jenkins

- Jenkins service for Continuous Integration
  - **Deploy** software packages on the systems in production
  - **Test** new Easyconfig files submitted by staff and users
  - **Check** regressions of Easyconfigs listed in production
  - **Update** production recipes in view of system upgrades
- Jenkins projects running with EasyBuild
  - **ProductionEB** builds the Easyconfigs once they are in production
  - **TestingEB** is triggered when a new pull request appears on Github
  - **UpdateEB** runs EasyBuild to update recipes and installed software
- Jenkins projects defined by **Pipelines**
  - Enhanced flexibility of the actions performed by Jenkins
  - **Jenkinsfile** script of each project is version controlled
  - The CI can **run in parallel** optimizing the available resources

CSCS production repository on GitHub

- **How to submit a pull request**
  - Add the EasyBuild configuration files to a **new branch** in your **fork**
  - The pull request must include **all the required dependencies**

- **Policy of pull requests**
  - The title **must match a supported system** (or the CI will fail)
  - System names are **enclosed in square brackets**
  - Dom and Piz Daint can test **-gpu** and **-mc** builds
    - [dom-gpu] NAMD (will build using **-gpu**)
    - [dom-mc] NAMD (will build using **-mc**)
    - [dom] NAMD (will use both **-gpu** and **-mc**)

- Jenkins project **TestingEB** tests the build of new recipes
  - Pipeline script at https://github.com/eth-cscs/production/tree/master/jenkins

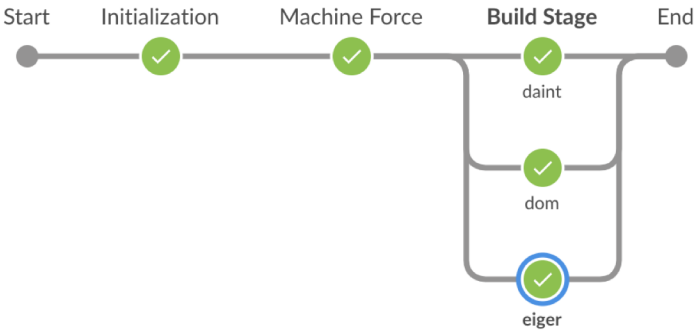
ProductionEB Pipeline

ProductionEB < 1600 | Pipeline | Changes | Tests | Artifacts | Logout

Branch: –
Commit: –

15m 50s
a day ago

Changes by Guilherme Peretti-Pezzi
Started by an SCM change

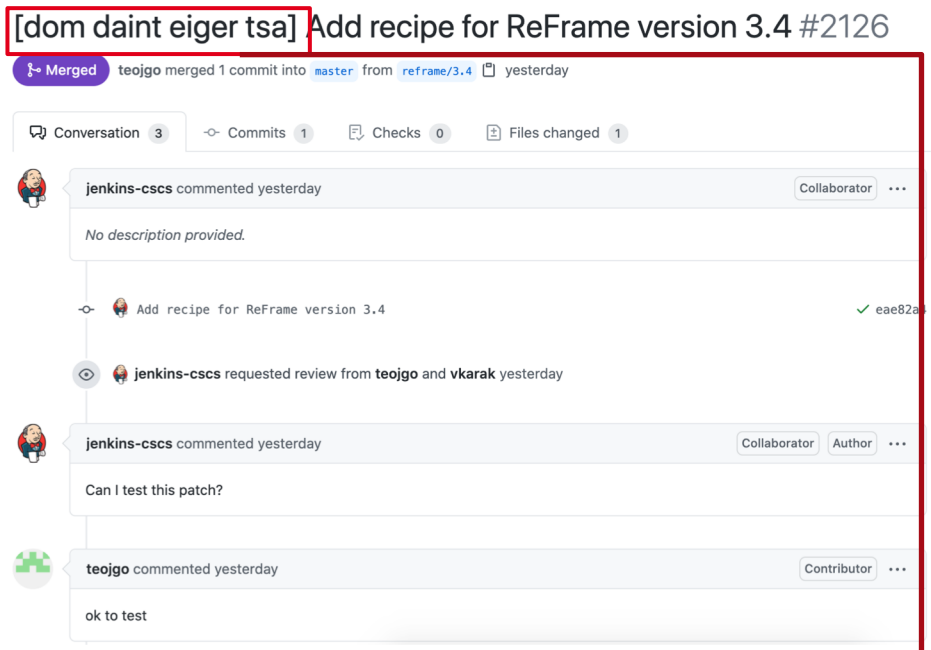


Build Stage / eiger - 15m 43s

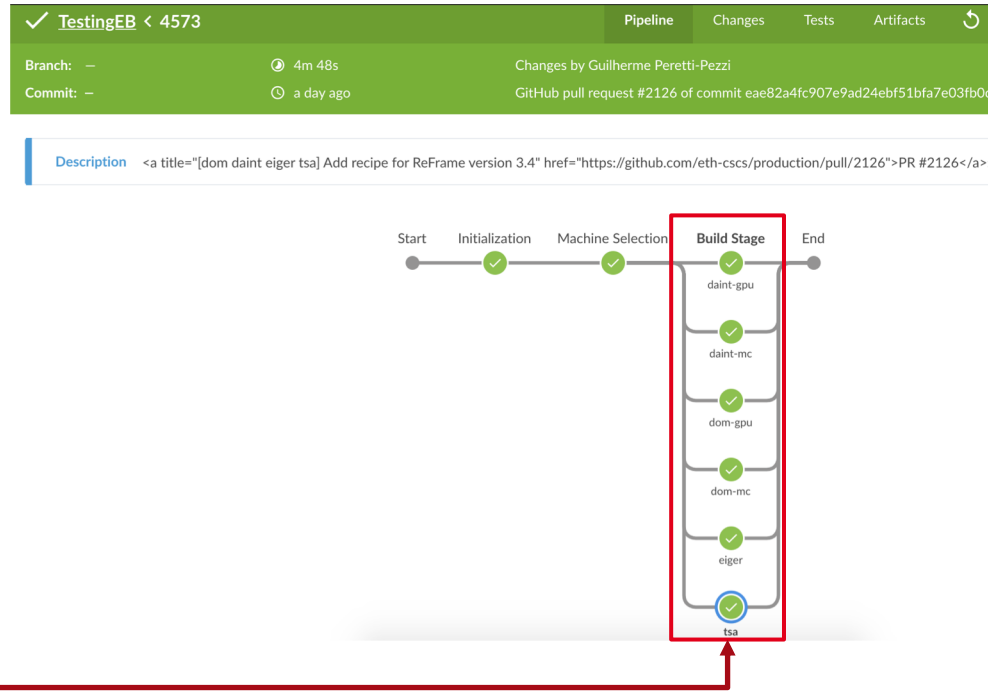
Check out from version control	3s
echo $PWD — Shell Script	<1s
#!/bin/bash -l echo $SCRATCH — Shell Script	<1s
#!/bin/bash -l echo $XDG_RUNTIME_DIR/build — Shell Script	<1s
#!/bin/bash -l echo /apps/eiger/UES/jenkins/1.3.2/20.10 — Shell Script	<1s
Shell Script	6m 52s

TestingEB Pipeline

Github Pull Request



Pipeline triggered on Jenkins



UpdateEB Pipeline

- Automated GitHub PR for successful updates (GitHub stage)

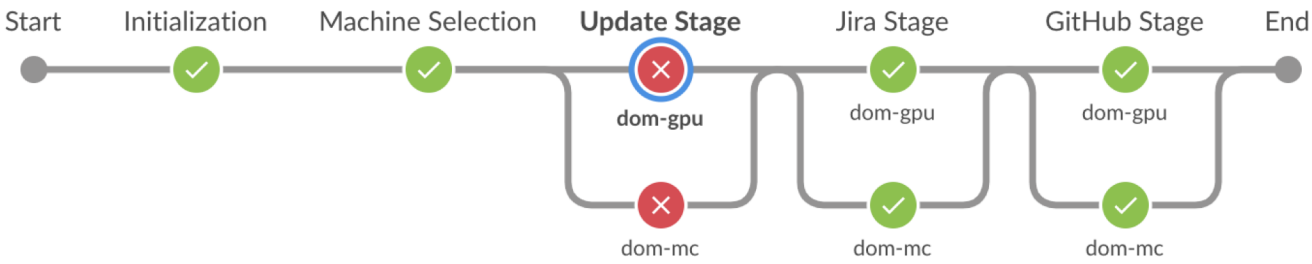
UpdateEB < 49 >

Pipeline | Changes | Tests | Artifacts

Branch: –
Commit: –

19h 15m 24s
11 days ago

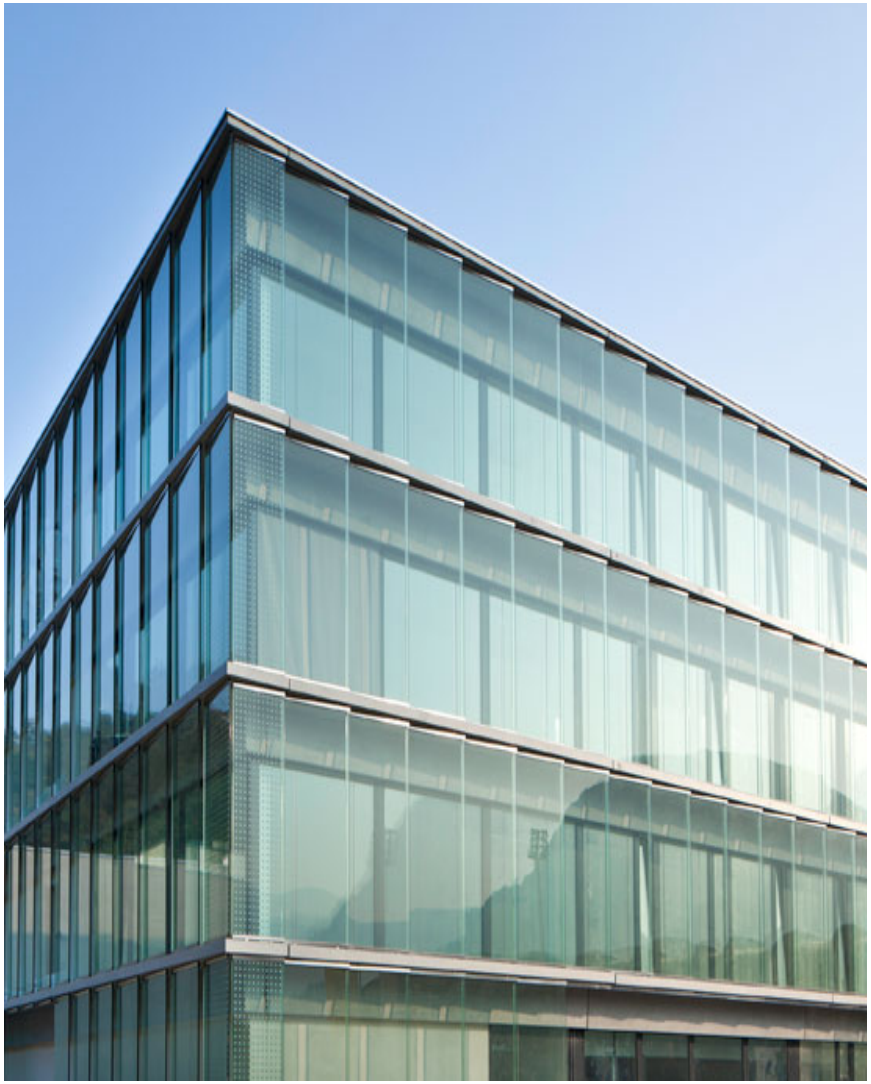
No changes
Started by user Luca



Update Stage / dom-gpu - 18h 48m 23s

- ✓ Check out from version control
- ✓ List of unuse paths: /apps/dom/UES/jenkins/7.0.UP02/gpu/easybuild/tools/modules/all /apps/dom/UES/jenkins/7.0.UP02/gpu/easybuild/modules/all
- ✗ Shell Script

Outline



- EasyBuild timeline @ CSCS
- CSCS HPC systems
  - Piz Daint
  - Alps
  - MeteoSwiss systems
- EasyBuild for CSCS Users
  - Custom User builds
  - Jenkins pipelines
- **UENV User Environments**
  - CLI and Slurm integration
  - The Stackinator tool

UENV User Environments: background

- User software installed on top of several layers
  - Cray Operating System (COS)
  - Cray Programming Environment (CPE)
  - CSCS software stack maintained by staff
- A change to one layer affects every layer above
  - COS and CPE updates often require rebuilding software
  - This applies both to CSCS software stack and users software
- CPE provides libraries and tools for many use cases
  - Once an issue is identified, the fix will come in a future release
  - Long latency between issue reporting and fix in production
  - New releases require extensive testing to check issue resolution

The **one size fits all model** does not scale with use cases

UENV User Environments: description

- The UENV User Environments approach
  - Login to a simpler environment
  - Minimal set of dependencies
    - COS + Slurm workload manager + runtime container + drivers
  - Load one or more user environments on demand
- Users can choose their environment
  - Classic CPE via modules: **module load cray**
  - CSCS-provided images for user environments
  - User-built user environments for advanced users
- Each environment is contained in a single file
  - Shared in an artifactory or stored on a filesystem
  - The environments are **independent of one another and of CPE**
  - The environments are built on top of the base-image - not the CPE

User Environments: tools

- Command Line Interface (CLI)
  - **squashfs-mount**: low level tool for mounting environments
  - **uenv**: a command line tool for interacting with environments
- **Slurm** plugin
  - manages loading UENV images on compute nodes
  - https://github.com/eth-cscs/slurm-uenv-mount
- **Stackinator** https://eth-cscs.github.io/stackinator
  - Tool for generating uenv images from a declarative recipe
  - Used by CSCS to build the user environments
  - Available for advanced users to build their own images
- GitHub repository with CSCS recipes
  - https://github.com/eth-cscs/alps-spack-stacks
  - CI/CD pipeline to build, test and deploy images

User Environments: mount points

- Mounted at **/user-environment**
  - Programming Environments
    - Compilers, MPI, libraries (e.g.: HDF5, FFTW, OpenBLAS…)
    - Can be application specific (e.g.: supporting ICON builds)
  - Application Environments
    - Provide applications, libraries and tools required to run them
    - CP2K, GROMACS, LAMMPS, NAMD, QuantumESPRESSO…
- Mounted at **/user-tools**
  - Debuggers: DDT
  - Profilers
  - Visualisation: ParaView, VisIt,…

User Environments: benefits

- Single image within a squashFS file
  - Managed in a registry/artifactory and not on the file system
  - Performance decoupled from the file system
- Defined by a simple declarative recipe
  - Key dependencies: **libfabric** and **Slurm** workload manager
  - The same environment can be rebuilt after system upgrades
- Small set of system dependencies
  - Only need rebuilding when **libfabric** or **Slurm** are changed

User Environments: getting started

- Get started with the latest development
  - **git clone https://github.com/eth-cscs/uenv.git**
  - **cd uenv && ./install –local**

- Test the status of the user environment
  - **uenv status**

- Load an environment
  - **uenv start $SCRATCH/gromacs-eiger.squashfs**

- Activate modules available in **/user-environment/modules**
  - **uenv modules --use**

Stackinator: quick start

- Install with **pip install stackinator** or from GitHub
  - Clone it from GitHub https://github.com/eth-cscs/stackinator.git
  - Run **bootstrap.sh** and install dependencies for stand-alone usage
  - Update your PATH with **<stackinator-install-path>/bin**

- Stackinator creates makefiles and spack configurations
  - equivalent to calling cmake or configure before running make
  - **stack-config --build $BUILD_PATH --recipe $RECIPE_PATH --system $SYSTEM_CONFIG_PATH**

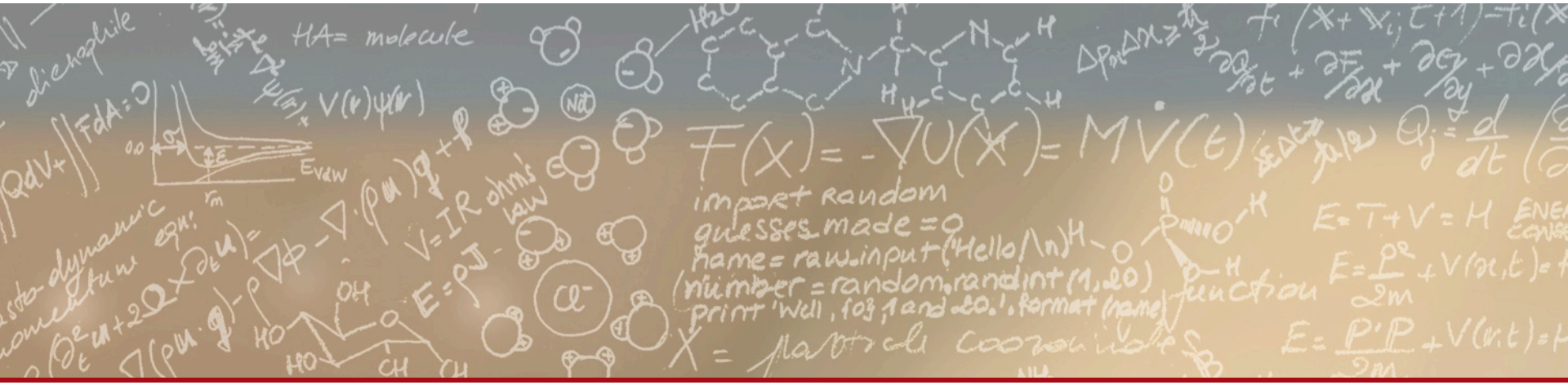
  - Configuration paths:
    - **BUILD_PATH** is the path where the build will be configured
    - **RECIPE_PATH** contains the recipe of the software stack
    - **SYSTEM_CONFIG_PATH** configuration of the target cluster

Stackinator: install

- The configuration generates a top-level Makefile
  - **env --ignore-environment PATH=/usr/bin:/bin:`pwd`/spack/bin make modules store.squashfs -j64**
  - The wrapper **env --ignore-environment** unsets environment variables to improve portability and reproducibility of the build
- The installation path is set by the configure step
  - The default location set in the store field of **config.yaml**
- make creates two software stacks in the build path
  - **store** sub-directory with the full software stack installation tree
  - **store.squashfs** compressed image of the of the store path
- The image can be mounted at runtime with UENV
  - squashfs-mount or the Slurm plugin or by a system-administrator

Useful links @ CSCS

- User Documentation
  - CSCS Knowledge Base https://docs.cscs.ch
  - User Portal still available https://user.cscs.ch
- CSCS production repository
  - https://github.com/eth-cscs/production
  - Mirror under the EasyBuilders GitHub repository
    - https://github.com/easybuilders/CSCS
- UENV User Environments
  - CSCS Knowledge Base article and https://eth-cscs.github.io/uenv
- Stackinator https://eth-cscs.github.io/stackinator



Thank you for your kind attention